IEEE Std 1609.2-2016 Guidance Note 4: HashedId

Paul Bottinelli, TrustPoint Innovation Technologies, 2016-07-15

1 Introduction

This note identifies an area in IEEE Std 1609.2-2016 that has been reported as potentially misleading or ambiguous and gives guidance on a recommended interpretation of the standard.

The potentially misleading or ambiguous text concerns the data structure HashedId.

This guidance note is structured as follows. Section 2 provides a summary of the issue. Section 3 provides the recommended interpretation. Section 4 provides the full text from the relevant sections of IEEE Std 1609.2-2016. Section 5 provides full replacement text that should be considered for incorporation into a revision or amendment of 1609.2.

2 Summary of Issue

IEEE Std 1609.2-2016 specifies three HashedId data structures, i.e., HashedId3, HashedId8 and HashedId10. The HashedId data structure corresponds to the truncation of the SHA-256 hash of an encoded piece of data. The HashedId3 (HashedId8 or HashedId10, respectively) is calculated by computing the SHA-256 hash of an encoded data structure, and then truncating the 32-byte hash to 3 bytes (8 or 10, respectively).

The potential confusion concerns how to truncate the output of the hash function. We will demonstrate in the example below the correct interpretation of all 3 types.

As an illustration, consider the text in IEEE Std 1609.2-2016 clause 6.3.25, which describes the HashedId3 data structure. It states that “The HashedId3 for a given data structure is calculated by calculating the SHA-256 hash of the encoded data structure and taking the low- order three bytes of the hash output. The low-order three bytes are the last three bytes of the 32-byte hash.”

By definition, the *low bytes* are the bytes that hold the least significant part of the data. Some confusion might arise if misinterpreting the wording *last three bytes*. Providing an example would certainly clarify any possible confusion.

Example: Computing the SHA-256 hash of the empty string, we get:

`SHA-256("") = e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855`

The HashedId3 of this hash corresponds to the three least significant bytes, i.e., the three rightmost bytes.

`HashedId3 = 52b855`

Similarly, the HashedId8 of this hash corresponds to

`HashedId8 = a495991b7852b855`

Finally, the HashedId10 corresponds to

`HashedId10 = 934ca495991b7852b855`

3. Recommended interpretation

The recommended interpretation of 1609.2 is to follow the straightforward meaning of the text in IEEE Std 1609.2-2016 and truncate the output of the SHA-256 hash by taking the low-order 3, 8, or 10 bytes.

4. Full text

6.3.25 HashedId3

```
HashedId3  ::= OCTET STRING (SIZE(3))
```

This data structure contains the truncated hash of another data structure. The HashedId3 for a given data structure is calculated by calculating the SHA-256 hash of the encoded data structure and taking the low-order three bytes of the hash output. The low-order three bytes are the last three bytes of the 32-byte hash.

Encoding considerations: If the data structure is a Certificate, the encoded Certificate which is input to the hash uses the compressed form for all elliptic curve points within the ToBeSignedCertificate and takes the *r* value of an ECDSA signature to be of type `x-only`. If the data structure is an Ieee1609Dot2Data containing a SignedData, the encoding takes the *r* value of an ECDSA signature to be of type `x-only`.

6.3.26 HashedId8

```
HashedId8  ::= OCTET STRING (SIZE(8))
```

This data structure contains the truncated hash of another data structure. The HashedId8 for a given data structure is calculated by calculating the SHA-256 hash of the encoded data structure and taking the low-

order eight bytes of the hash output. The low-order eight bytes are the last eight bytes of the 32-byte hash.

Encoding considerations: If the data structure is a Certificate, the encoded Certificate which is input to the hashusesthecompressedformforallellipticcurvepointswithintheToBeSignedCertificate andshalltake the *r* value of an ECDSA signature to be of type `x-only`. If the data structure is an Ieee1609Dot2Data containing a SignedData, the encoding takes the *r* value of an ECDSA signature to be of type `x-only`.

6.3.27 HashedId10

```
HashedId10 ::= OCTET STRING (SIZE(10))
```

This data structure contains the truncated hash of another data structure. The HashedId10 for a given data structure is calculated by calculating the SHA-256 hash of the encoded data structure and taking the low- order ten bytes of the hash output. The low-order ten bytes are the last ten bytes of the 32-byte hash.

Encoding considerations: If the data structure is a Certificate, the encoded Certificate which is input to the hash uses the compressed form for all elliptic curve points within the ToBeSignedCertificate and takes the r value of an ECDSA signature to be of type `x-only`. If the data structure is an Ieee1609Dot2Data containing a SignedData, the encoding takes the *r* value of an ECDSA signature to be of type `x-only`.

5. Proposed replacement text

When the standard is to be revised, we recommend to add the example provided above.

6.3.25 HashedId3

```
HashedId3  ::= OCTET STRING (SIZE(3))
```

This data structure contains the truncated hash of another data structure. The HashedId3 for a given data structure is calculated by calculating the SHA-256 hash of the encoded data structure and taking the low-order three bytes of the hash output. The low-order three bytes are the ~~last three bytes of the 32-byte hash~~ least significant three bytes of the 32-byte hash when represented as an integer. See Example below.

Encoding considerations: If the data structure is a Certificate, the encoded Certificate which is input to the hash uses the compressed form for all elliptic curve points within the ToBeSignedCertificate and takes the *r* value of an ECDSA signature to be of type `x-only`. If the data structure is an Ieee1609Dot2Data containing a SignedData, the encoding takes the *r* value of an ECDSA signature to be of type `x-only`.

Example: Consider the SHA-256 hash of the empty string:
`SHA-256("") =`
`e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b78`**`52b855`**

The HashedId3 of this hash was highlighted above and corresponds to the following:

HashedId3 = 52b855.

6.3.26 HashedId8

```
HashedId8  ::= OCTET STRING (SIZE(8))
```

This data structure contains the truncated hash of another data structure. The HashedId8 for a given data structure is calculated by calculating the SHA-256 hash of the encoded data structure and taking the low-order eight bytes of the hash output. The low-order eight bytes are the ~~last eight bytes of the 32-byte hash~~ least significant eight bytes of the 32-byte hash when represented as an integer. See Example below.

Encoding considerations: If the data structure is a Certificate, the encoded Certificate which is input to the hashusesthecompressedformforallellipticcurvepointswithintheToBeSignedCertificate andshalltake the *r* value of an ECDSA signature to be of type `x-only`. If the data structure is an Ieee1609Dot2Data containing a SignedData, the encoding takes the *r* value of an ECDSA signature to be of type `x-only`.

Example: Consider the SHA-256 hash of the empty string:
SHA-256("") =
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934c**a495991b7852b855**

The HashedId8 of this hash was highlighted above and corresponds to the following:
HashedId8 = a495991b7852b855.

6.3.27 HashedId10

```
HashedId10 ::= OCTET STRING (SIZE(10))
```

This data structure contains the truncated hash of another data structure. The HashedId10 for a given data structure is calculated by calculating the SHA-256 hash of the encoded data structure and taking the low- order ten bytes of the hash output. The low-order ten bytes are the ~~last ten bytes of the 32-byte hash~~ least significant ten bytes of the 32-byte hash when represented as an integer. See Example below.

Encoding considerations: If the data structure is a Certificate, the encoded Certificate which is input to the hash uses the compressed form for all elliptic curve points within the ToBeSignedCertificate and takes the r value of an ECDSA signature to be of type `x-only`. If the data structure is an Ieee1609Dot2Data containing a SignedData, the encoding takes the *r* value of an ECDSA signature to be of type `x-only`.

Example: Consider the SHA-256 hash of the empty string:
SHA-256("") =
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b**934ca495991b7852b855**

The HashedId10 of this hash was highlighted above and corresponds to the following:
HashedId10 = 934ca495991b7852b855.